Please use C++ (or Java) for all programming questions.

1. **(10 %)** Please convert the following infix expressions to postfix:
(a) (5 %) a * (b – c) / d
(b) (5 %) a + (b * c – d) * (e + f / g)

2. **(15 %)** Write a method called **makeList** in a **SList** class with a tail. The **makeList** function takes an array counts of **int**, constructs a singly-linked list, and associates its head and tail pointers to the corresponding **SListNode** nodes. In the constructed list, the first **counts[0]** items are the number **counts[0]**, the next **counts[1]** items are the number **counts[1]**, etc. For example, if the input array is {1 3 2}, then the output linked list is: **head**→ [1] → [3] → [3] → [3] → [2] → [2] ←**tail**. Define the **SList** and **SListNode** classes to complete the **makeList** method. When it is done, the **size** of nodes in **SList** should also be updated accordingly.

3. **(10 %)** Examine the following statements. If it is correct, prove it. If it is incorrect, show the contradiction.
(a) (5 %) $1/(n-50) \in O(n)$
(b) (5 %) $n^2/(n+\log n) \in \Theta(n^2)$

4. **(10 %)** Show the sequence of swaps by which the method **bottomUpHeap** converts the following array into a min heap. Redraw the array once for each swap.

	7	2	6	9	3	5	1	8

5. **(15 %)** Write a recursive method **preorderElementAt(int k)** that returns the kth element in the preorder listing of the subtree in a binary tree. For example, if $k = 1$, return the element in the root node. The method should run in $O(d)$ time, where d is the depth of the subtree. If the subtree doesn't have k nodes, return null. Define all necessary classes and associated fields to accomplish the function.

6. **(12 %)** A graph G is **bipartite** if its vertices can be partitioned into two sets X and Y such that every edge in G has one end vertex in X and the other in Y. Describe (not code) an efficient algorithm for determining if an undirected graph G is bipartite (without knowing the sets X and Y in advance).

7. **(16 %)** Sorting.
(a) (8 %) Trace the bubble sort as it sorts the following array into ascending order:
15 4 2 27 10
(b) (8 %) Is the bucket-sort algorithm in-place? Why or why not?

8. **(12 %)** Suppose you insert a set of keys {4, 6, 12, 15, 3, 5} into an initially empty 2-3-4 tree in that order. Now, you do the same insertion to an empty 2-3-4 tree with the same keys but a different order, say {12, 3, 6, 4, 5, 15}. Can these two trees have the same structure? Draw both trees to justify your answer.

試題隨卷繳回